Data: waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

Goal: construct a parametric bootstrap confidence interval for the coefficient of variation of waiting time to next eruption for the Old Faithful geyser using the rejection sampling method.

Part I. Estimating the parameters of a distribution.

```
library(datasets)
library(ggplot2)
str(faithful) # structure of the data, N = 272
```

```
## 'data.frame': 272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...
```

```
head(faithful) # first observations of the Old Faithfull Geyser data
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

```
# make a histogram of the waiting time (figure 1)
df.wait <- data.frame(faithful$waiting)
p <- ggplot(df.wait, aes(x=faithful$waiting)) + geom_histogram(aes(y=..count..), binwidth=4)
p <- p + labs(title = "Histogram of the waiting time") + ylab("Count")
p <- p + xlab("Minutes")
p
```

```
suppressMessages(library(mixtools))
x.w <- faithful$waiting

# estimate the mixture model parameters with two normals using the EM-algorithm
x.mix <- normalmixEM(x.w)
```

```
## number of iterations= 19
```

```
x.mix[c("lambda", "mu", "sigma")] # get mixing proportions, means and std deviations
```

```
## $lambda
```

Figure 1: The waiting time between eruptions has a binormal distribution with two peaks around 50 and 80 minutes.

```
## [1] 0.3608852 0.6391148
##
## $mu
## [1] 54.61482 80.09105
##
## $sigma
## [1] 5.871195 5.867753


plot(x.mix, which = 2, breaks = 20) # plot density components (figure 2)
```

Part II. Rejection sampling.

```
# get an envelope

# target density of the mixture of normals
f <- function(q)
{
    d1 <- dnorm(q, mean=x.mix$mu[1], sd=x.mix$sigma[1])
    d2 <- dnorm(q, mean = x.mix$mu[2], sd = x.mix$sigma[2])
    x.mix$lambda[1]*d1 + x.mix$lambda[2]*d2 # sum of mixing proportions times by densities
}

# uniform proposal density (p.s. normal density can be used instead)
g <- function(q, a, b)
{
    dunif(q, a, b);
}
```

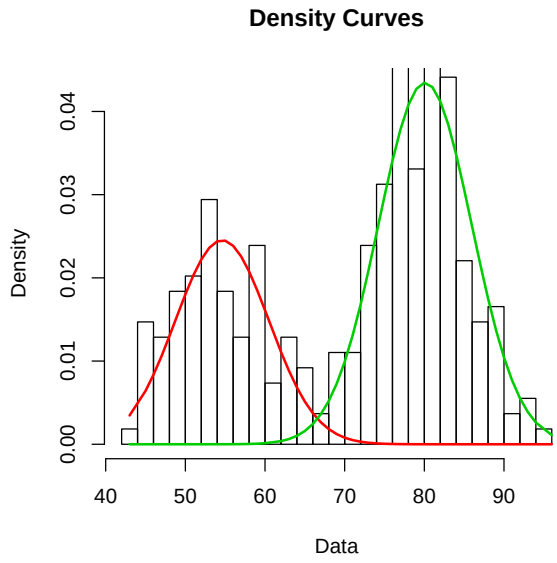

Figure 2: On average, people are waiting for the next geyser eruption either 55 or 80 minutes. 36% of observations distributed N(54.615, 5.871) and 64% of observations distributed N(80.091, 5.868).

```
# since waiting time range is approx (30,100) mins
q <- seq(30, 100, length.out = 1000)

M <- 3.2 # scale factor
envelope <- g(q, 30, 100) * M # envelope = proposal_distr * scale_factor
mixture_density <- f(q) # mixture density

# plot envelope (figure 3)
plot(q, envelope, xlim=range(q), ylim=c(0,0.10), type="l", col = "blue", lwd=3,
     main="The mixture density and U(30,100) envelope")
points(q, mixture_density, type="l", col = "red", lwd=1) # add mixture density to the plot
```

```
# Rejection sampling
# since N = 272, we need to have 272 accepted samples
N <- 272
# create empty vectors
accept <- c()
y <- c()
i <- 0 # counter
# loop until we have N = 272 accepted samples
while (length(accept[which(accept=="Yes")]) != N){
  i <- i + 1
  y[i] <- runif(1,30,100) # draw y from proposal distribution g(y) = U(30,100)
  u <- runif(1, 0, 1) # draw u from Uniform(0,1)
  # accept if u < f(y)/(M*g(y))
  if (u  < f(y[i]) / (M * g(y[i], 30, 100)) ){
    accept[i] <- 'Yes'
  }
```

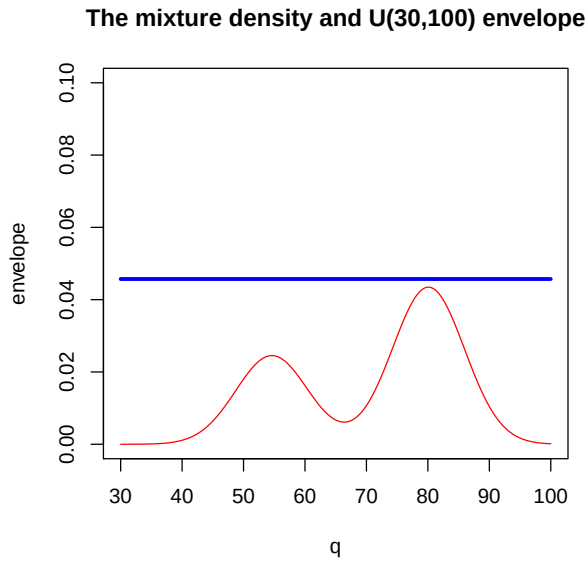

Figure 3: The uniform envelope function e(x) is greater than density of the mixture distribution f(x) as shown on the plot.

```
  else { accept[i] <- 'No'} # else reject
}

# put into a data.frame for plotting
sam <- data.frame(y, accept=factor(accept, levels = c("Yes","No")))
# plot a stacked histogram (figure 4)
p <- ggplot(sam, aes(x = y))
p <- p + geom_histogram(aes(fill = accept))
print(p)


## `stat_bin()` using `bins = 30`.  Pick better value with `binwidth`.
```

Part III. Parametric bootstrap.

```
# function to sample random deviates using the rejection sampling method
reject.sample <- function(N, g, f, M){
  R <- N^2
  y <- runif(R,30,100)     # sample from enveloping function
  accept <- rep("No", R)   # initialize samples as "No" accept
  u <- runif(R, 0, 1)      # sample from uniform distribution
  accept[ u < f(y) / (M * g(y)) ] <- "Yes"
  samp <- data.frame(y, accept = factor(accept, levels = c("Yes")))
  samp <- na.omit(samp) # delete all NAs
  rand_samp <- samp[sample(nrow(samp), N), ] # select N random samples from all rejected samples
  return(rand_samp)
}
```

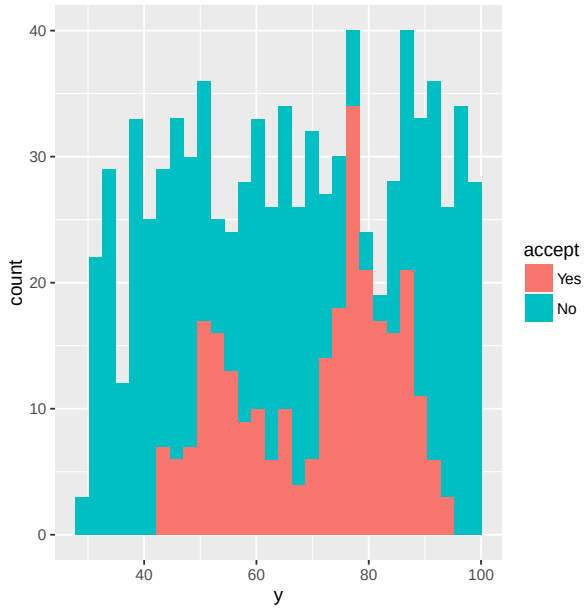

Figure 4: The histogram of the accepted and rejected samples.

```
proposal.dens  <- function(x) g(x, 30, 100)     # proposal density
target.dens <- function(x) f(x)              # target density

res.sample <- reject.sample(N=272, g=proposal.dens, f=target.dens, M=3.2)
```

```
# parametric bootstrap
R <- 1e4 # bootstrap sample size

# draw N=272 sample obtained by rejection sampling
values <- res.sample$y
n <- sqrt((N - 1) / N) # correction factor

# sample summaries
mu.hat <- mean(values) # mean
sd.hat <-n*sd(values) # sd

# the CV is a standardized measure of dispersion of a probability distribution
cv.hat <- 100 * sd.hat / mu.hat # estimate of the CV in %

# draw a matrix of samples
sample.m <- matrix(rnorm(R*N, mean = mu.hat, sd = sd.hat), nrow = R)

# get bootstrap cv's
cv.bs <- 100 * n * apply(sample.m, 1, sd) / apply(sample.m, 1, mean)

cv.sorted <- sort(cv.bs)
dat.cv <- data.frame(cv.sorted)

# 0.025th and 0.975th quantile gives equal-tail bootstrap CI
```

```
CI.bs <- c(cv.sorted[round(0.025*R)], cv.sorted[round(0.975*R+1)])
# The 95% parametric bootstrap CI is
CI.bs


## [1] 16.58139 19.71376
```

```
p <- ggplot(dat.cv, aes(x=cv.sorted)) # (figure 5)
p <- p + geom_histogram(aes(y = ..density..), binwidth=0.5)
p <- p + labs(title = "Parametric bootstrap of CV")
p <- p + geom_vline(aes(xintercept=cv.hat), colour="#BB0000", linetype="solid")
p <- p + geom_vline(aes(xintercept=CI.bs[1]), colour="#00AA00", linetype="longdash")
p <- p + geom_vline(aes(xintercept=CI.bs[2]), colour="#00AA00", linetype="longdash")
p.param <- p + xlab(expression(paste("red = ", hat(cv), ", green = bootstrap CI")))
print(p.param)
```

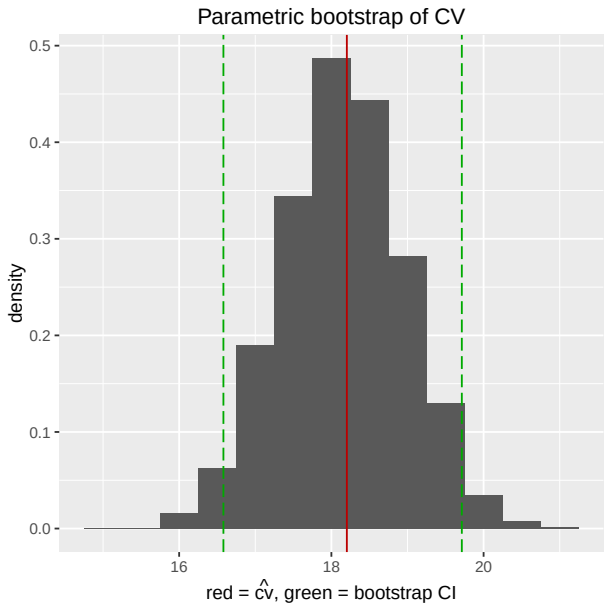

Figure 5: The 95% parametric bootstrap CI using the rejection method of sampling from the mixture distribution.